

(43)Date of publication of application : 07.06.2002

G06F 9/46
G06F 12/08

(71)Applicant : TOSHIBA CORP

(72)Inventor : ASANO SHIGEHIRO
SAITO MITSUO

(57)Abstract:

SOLUTION: A virtual thread number is allocated to a thread newly started on hardware threads 3 and 13 or the like. A register file cache 5 caches the contents of the registers of the threads, with the virtual thread identification numbers and register numbers used as keys. An issue section 4 performing issue control gains access to the register file cache 5 about the command to be issued, with the relevant thread identification number and the register identification number used as keys. When the cache is hit, it is determined that the command can be issued at least on the register. When the cache is missed, the relevant data are transferred between the register file cache 5 and a context saving memory 11.

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

THIS PAGE BLANK (USPTO)

(5) InCl_3	濃度配分	P I	7-10-1 (参考)
G 0 6 F	9/46	3 4 0	5 B 0 5
		3 1 3	3 1 3 D
12/08		5 5 9	5 B 9 8
		5 5 9 Z	
	5 5 5		5 5 5

未請求 請求項の数18 OL (全13頁)

(21) 出版番号	特製2000-355236(F2000-355238)	(71) 出版人	000003078 株式会社東芝
(22) 出版日	平成12年11月22日(2000.11.22)		東京都港区芝浦一丁目1番1号

最終頁に宛て

(54) 【発明の名称】 仮想マルチスロットプロセッサ及びスロット実行方法

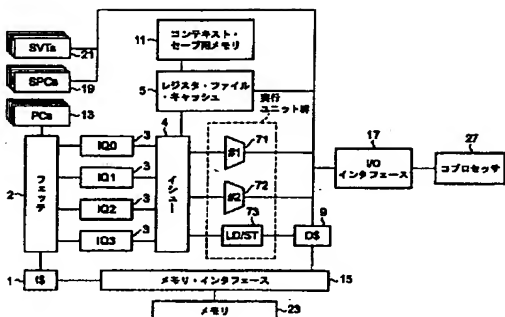
(57) 【要約】

【課題】 ハードウェアによる制限を受けずに多数のスレッドを扱うことのできる仮想マルチスレッドプロセッサを提供すること。

【解決手段】 ハードウェア・スレッド(3、13等)

上で新規起動されたスレッドに、仮想スレッド番号を割り当てる。レジスタ・フリアル・キャッシュ5は、仮想スレッドの各レジスタの内容をキャッシュする。発行待ち状態にあるスレッドが識別番号およびレジスタ識別番号をキープを行うイシュー部4は、発行対象となる命令について、該当するスレッド識別番号およびレジスタ識別番号をキーとしてレジスタ・フリアル・キャッシュをアクセスできるようにしている。この場合にも、少なくともレジスタに関しては、ヒットした場合には、少なくともレジスタに関してのみ、命
令の発行が可能であると判断する。キャッシュとミスの場合には、レジスタ・フリアル・キャッシュ5とミニステキスト・セーフ用メモリ1との間で該当するデータの転送が行われる。

（以下省略）



【特許請求の範囲】

【請求項1】複数のスレッドを同時に実行できる仮想マ
ルチスレッドプロセッサであって、

レップを特定する仮想スレップ識別番号を割り当てるための手段と、

システム段と、
各レコードの各レコードの内容をキャッチするためのキャ
ッチ番号を特定するレコード番号をキーとして各レ
コードの各レコードの内容をキャッチするためのキャ
ッチ番号を特定するレコード番号をキーとして各レ
コードの各レコードの内容をキャッチするためのキャ
ッチ番号を特定するレコード番号をキーとして各レ

前記プロセッサ上にある各スレッドの命令の発行制御を
行う発行制御手段とを備え、

前記発行制御手段は、発行対象となる命令について該当する仮想レジスタ識別番号およびレジスタ識別番号をキーとして前記キャッシュ手段をアクセスし、ヒットし且つ発行する命令の依存性が解決している場合と、少なくともレジスタに関連しては該命令の発行は可能であると判断することと特徴とする仮想マルチスレッドプロセッサ。

【請求項2】 前記発行制御手段は、発行可能な命令が複数ある場合、且つ空いている演算ユニット数の制限のためにそれぞれ異なる発行速度を有する命令を、発行可能な命令のうちのすべてを発行することができない場合には、所定の優先度に基づいて選択した一部の命令のみ発行することと特徴とする請求項1に記載の仮想マルチスレッドプロセッサ。

【請求項3】 前記発行制御手段により前記発行対象となる命令について前記仮返レップ識別番号および前記レップスタ識別番号をキーとして前記キャッシュ手段がフランクセンスされ、キャッシュ・ミスが発生した場合に、該キャッシュ手段と所定のメモリとの間で該当するデータの転送を行うことを特徴とする請求項1に記載の仮返レップレッププロセッサ。

【請求項4】 前記プロセッサ上で実行中のスレッドがレージングの長いハードウェア資源に対して要求を出した場合には明示的に該スレッドを待避させる第1の処理と、前記待避中の前記スレッドが出した前記要求に対する前記所定のハードウェア資源からの応答があった後に該応答に

対応する要求を出したスレップを復帰すべきと判断され、場合によってはスレップを復帰させる第2の処理とを行うための処理手段を更に備えたことを特徴とする請求項1に記載の仮想マルスレッププロセッサ。

前記処置手段は、前記第2の処理において、少なくとも一つのスレッドを起動するのに必要な資源の空きがあり、かつ、新規に起動するべき他のスレッドが前記応答があったスレッドのいずれかである場合に、前記応答があったスレッドのいずれかであることと判断することを特徴とする請求項4に記載の仮想マシンスレッドプロセッサ。

【請求項6】前記仮想スレッド識別番号に対応して該仮想スレッド識別番号のスレッドが前記応答を返されて復元可能な状態にあるかそれ以外の状態にあるかを少なくとも

とも示す情報を保持する所定のレジスタを更に備え、

前記処理手段は、前記第2の処理において、前記所定のレジスタを参照することによって、復帰可能なスレッドを特定することを特徴とする請求項2に記載の仮想マシンスレッドプロセッサ。

【請求項7】前記処理手段における前記第2の処理の少なくとも一部を、特別なスレッドによって実行することなくとる請求項4に記載の仮想マルチスレッドプロセッサ。

【請求項8】 前記記録手段からの前記所定のユーザデータ資源への要求に、該スレッドに対する該所定のユーザデータ資源を伴わせることと、該要求を出した該スレッドの必須スレッド識別番号を伴えることによって、該必須スレッド識別番号を伴えることにより、該必須スレッド識別番号を伴えることとなる要求を出したスレッドを対応付けることを特徴とする請求項4に記載の仮想マルチスレッドプロセッサ。

【請求項9】前記所定のメモリは、前記プロセッサが命令によりアクセスするメモリとは別に専用のメモリとして構成されることを特徴とする請求項1に記載のマルチスレッドプロセッサ。

【請求項10】複数のスレッドを同時に実行できる仮想マルチスレッドプロセッサにおけるスレッド実行方法であって、

前記プロセッサ上で新規に起動されたスレッドに、設定されたスレッドを割り当て、各スレッドを特定する仮想スレッド識別番号および各スレッドを特定するレジスタ識別番号をキーとして各スレッドの各レジスタの内容をキャッシュ手段にキャッシュし、

行う際に、発行対象となる命令について該当する仮型スロット識別番号およびレジスタ識別番号をキーとしてまずレジスタ識別番号を手渡すアクセス、ヒットし且つ先行している命令の依存性が解決している場合に、少なくともレジスタ識別番号に関しては該命令の発行が可能であると判断することとを特徴とするスレッド実行方法。

【請求項11】複数のスレッドを同時に実行できる仮想マルチスレッドプロセッサであって、前記マルチスレッド上で新規に起動されたスレッドに、該スレッドを特定する仮想スレッド識別番号を割り当てるための手段と、少なくとも一つの同期用カウンタと、

数に設定するための手段とを備え、同期をとるべき各々のスレップにおいて同期をとるため、同期命令に達した際には、前記同期用カウンタをデクrementし、デクrementされた後の前記同期命令を発行したスレップの値が1にならない場合、当該同期命令を待避させ、デクrementを復帰可能でない状態として待避させ、デクrement

3

ントされた後の前記同期用カウンタの値が0になれば、当該同期命令を発行したスレップは、先に同期命令に達し復帰可能でない状態として待避されたスレップの状態を、復帰可能な状態に変更することを特徴とする仮想マルチスレップ状態。

【請求項12】 復帰可能な状態として待避されたスレップを、所定の復帰条件が成立した場合に復帰させる処理を行うための手段を更に備えたことを特徴とする請求項11に記載の仮想マルチスレッププログラム。

【請求項13】 前記同期命令は前記同期用カウンタを識別する情報の一に記載のマルチスレッププログラムであることを特徴とする請求項11に記載の仮想マルチスレッププログラム。

【請求項14】 前記同期用カウンタを識別する情報として前記スレップを識別する仮想スレップ識別番号を用いることを特徴とする請求項13に記載のマルチスレッププログラム。

【請求項15】 各スレップを特定する仮想スレップ識別番号および各スレップを特定するレジスタ識別番号をキーとして各スレップの各レジスタの内容をキャッシュするためのキャッシュ手段を更に備え、前記同期用カウンタと前記キャッシュ手段を通してアクセスできるようにしたことを特徴とする請求項11に記載のマルチスレッププログラム。

【請求項16】 複数のスレップを同時に実行できる仮想マルチスレッププログラムにおけるスレップ実行方法であって、前記プログラム上で新規に起動されたスレップに、該スレップを特定する仮想スレップ識別番号を割り当て、同期用カウンタの値を、同期をとるべきスレップの数に設定し、

同期をとるべき各々のスレップにおいて同期をとるための同期命令に達した際には、前記同期用カウンタをデクリメントし、デクリメントされた後の前記同期用カウンタの値が0にならなければ、当該同期命令を発行したスレップを復帰可能でない状態として待避させ、デクリメントされた後の前記同期用カウンタの値が0になれば、当該同期命令を発行したスレップは、先に同期命令に達し復帰可能でない状態として待避されたスレップの状態を、復帰可能な状態に変更することを特徴とする請求項15の実行方法。

【発明の詳細な説明】

【0001】
【発明の属する技術分野】 本発明は、複数のスレップを実行可能な仮想マルチスレッププログラム及びスレップ実行方法に関する。

【0002】
【従来の技術】 一つのプログラム上で複数のスレップを実行するマルチスレッププログラムでは、CPU内部の演算器やメモリのリーチレンジを越えるための技術として広く知られている。

4

【0003】 しかしながら、従来のマルチスレッププログラムでは、レジスタファイルなどのハードウェア資源をスレップごとに用意することで複数のスレップを実装するものであるため、実装できる（すなわちユーザが使用できる）スレップの数は、レジスタファイルなどのハードウェア資源の限界から限定されたものであった。

【0004】 しかしながら、一度に可能になるスレップの数を例えば4程度に制限し、ハードウェア量が膨大になるのを防いでいた。これをプログラムの立場からみると、限定されたスレップの数はプログラミミングの自由度に対して大きな制約を課していることになり、問題であった。

【0005】
【発明が解決しようとする課題】 以上のように、従来のマルチスレッププログラムでは、スレップの数がハードウェアの資源の限界から限定され、プログラミミングの自由度に対して大きな制約を課していた。

【0006】 本発明は、上記事情を考慮してなされたもので、ハードウェアによる制約を受けずに多数のスレップを扱うことのできる仮想マルチスレッププログラムを提供することを目的とする。

【0007】 また、マルチスレップで記述されているプログラムでは、スレップ間の同期をとることが重要である。

【0008】 本発明は、効率的な同期手段を持つ仮想マルチスレッププログラムを提供することを目的とする。

【0009】
【課題を解決するための手段】 本発明は、複数のスレップを同時に実行できる仮想マルチスレッププログラムであって、前記プログラム上で新規に起動されたスレップに、該スレップを特定する仮想スレップ識別番号を割り当てるための手段と、各スレップを特定する仮想スレップ識別番号および各スレップの各レジスタの内容をキャッシュするためのキャッシュ手段と、前記プログラムにある各スレップの命令の発行制御を行う発行制御手段とを備え、前記発行制御手段は、発行対象となる命令について該当する仮想スレップ識別番号およびレジスタ識別番号をキーとして前記キャッシュ手段をアクセスし、スレップと先行する命令の依存性が解決している場合に、少なくとも1つのスレップに対しては該命令の発行が可能であると判断することを特徴とする。

【0010】 好ましくは、前記発行制御手段は、発行可能な命令が複数あり且つ受けている演算ユニット数の制約のためにそれらのすべてを発行することができない場合に、所定の基準で選択された一部の命令のみを発行するようにしてもよい。好ましくは、前記発行制御手段により前記発行対象となる命令について前記仮想スレップ識別番号および前記レジスタ識別番号をキーとして前記キャッシュ手段がアクセスされ、キャッシュ・ミスが発生した場合に、該キャッシュ手段と所定のメモリとの間で該当するデータの転送を行うようにしてもよい。好ましくは、前記プログラム上で実行中のスレップがリーチレンジの長いハードウェア資源に対して要求を出した場合には、明示的に該スレップを待避させる第1の処理と、待避中の前記スレップが出した前記要求に対する前記所定のハードウェア資源からの応答があった後に該依存性に対応する要求を出したスレップを復帰すべきと判断される場合に該スレップを復帰させる第2の処理とを行うための処理手段を更に備えるようにしてもよい。

【0011】 また、本発明は、複数のスレップを同時に実行できる仮想マルチスレッププログラムであって、前記プログラム上で新規に起動されたスレップに、該スレップを特定する仮想スレップ識別番号を割り当てるための手段と、同期をとるべきスレップの数に設定するための手段とを備え、同期をとるべき各々のスレップにおいて同期をとるための同期命令に達した際には、前記同期用カウンタをデクリメントし、デクリメントされた後の前記同期用カウンタの値が0にならなければ、当該同期命令を発行したスレップを復帰可能でない状態として待避させ、デクリメントされた後の前記同期用カウンタの値が0になれば、当該同期命令を発行したスレップは、先に同期命令に達し復帰可能でない状態として待避されたスレップの状態を、復帰可能な状態に変更することを特徴とする。

5

生した場合に、該キャッシュ手段と所定のメモリとの間で該当するデータの転送を行うようにしてもよい。好ましくは、前記プログラム上で実行中のスレップがリーチレンジの長いハードウェア資源に対して要求を出した場合には、明示的に該スレップを待避させる第1の処理と、待避中の前記スレップが出した前記要求に対する前記所定のハードウェア資源からの応答があった後に該依存性に対応する要求を出したスレップを復帰すべきと判断される場合に該スレップを復帰させる第2の処理とを行うための処理手段を更に備えるようにしてもよい。

【0011】 また、本発明は、複数のスレップを同時に実行できる仮想マルチスレッププログラムであって、前記プログラム上で新規に起動されたスレップに、該スレップを特定する仮想スレップ識別番号を割り当てるための手段と、同期をとるべきスレップの数に設定するための手段とを備え、同期をとるべき各々のスレップにおいて同期をとるための同期命令に達した際には、前記同期用カウンタをデクリメントし、デクリメントされた後の前記同期用カウンタの値が0にならなければ、当該同期命令を発行したスレップを復帰可能でない状態として待避させ、デクリメントされた後の前記同期用カウンタの値が0になれば、当該同期命令を発行したスレップは、先に同期命令に達し復帰可能でない状態として待避されたスレップの状態を、復帰可能な状態に変更することを特徴とする。

【0012】 好ましくは、復帰可能な状態として待避されたスレップを、所定の復帰条件が成立した場合に復帰させる処理を行うための手段を更に備えるようにしてもよい。

【0013】 なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

【0014】 本発明では、レジスタファイルを、仮想スレップ識別番号（VTID）およびレジスタ識別番号（RID）をキーとして引かれるキャッシュとして構成する。プログラム上の資源に割り当てる仮想スレップ識別番号（VTID）を適宜切り替えることによって、実際にハードウェア上に存在するスレップより、仮想的に多くのスレップが実行できるようにみせることができる。

【0015】 また、一般にプログラムによって使用されるレジスタの数は、用意されているレジスタの数より少ないことから、スレップを切り替えるとき、レジスタファイルをすべてセーブ（Save）およびリスート（Restore）するかわりに、必要なものだけセーブおよびリスートすることができ。

【0016】 また、本発明では、複数のスレップを同時に実行できるSMT（Simultaneous Multi Thread）プログラムにおいて、キャッシュ

6

されたレジスタファイルにミスしたときは、ミスしたレジスタを使用する命令の発行が行われないだけなので、プログラム全体がストールすることを回避できる。つまり、SMTプログラムにおける、レジスタによって、レジスタファイルにミスしたスレップは、レジスタが利用可能でない、すなわち発行可能でないとして判定され、他のスレップから発行可能な命令が発行されることになる。

【0017】 本発明によれば、スレップの待避・復帰が可能とすることによって、プログラムにハードウェアで用意されたより多くのスレップを見せることが可能になり、プログラミミングの自由度が増す。また、その際には、マルチスレッププログラム上のレジスタの内容などをメモリ上にセーブおよびリスートする回数を少なくし、実行時間を短縮することができる。

【0018】 また、本発明では、仮想スレップ番号（VTID）ごとに同期機構を実現するためのカウンタを設ける。このカウンタは、同期にかかわる複数のスレップが同期命令によってこのカウンタをデクリメントするよう作用する。なお、カウンタの初期値は、同期にかかわるスレップの数が設定されている。デクリメントの結果、カウンタの値が0でなくなったスレップは、Sleep状態に入る。デクリメントの結果、カウンタが0であったスレップは、同期処理ルーチンを実行することで、最後に同期命令を実行したスレップと他のスレップの同期が取れることになる。

【0019】 このように、本発明によれば、効率的にスレップ間の同期をとることができる。

【0020】
【発明の実施の形態】 以下、図面を参照しながら発明の実施の形態を説明する。

【0021】（第1の実施形態） まず、本発明の第1の実施形態について説明する。

【0022】 図1は、本実施形態に係る仮想マルチスレッププログラムの構成を示す。

【0023】 なお、図1では、本発明をSMT（Simultaneous Multi Thread）プログラムに適用した例として、In-Order実行モデルを示している。

【0024】 まず、ハードウェア・スレップの概念について説明しておく。ハードウェア・スレップは、インストラクション・キュー（図1では3）に接続されている各プログラムのカウンタ（図1では13）に対応した概念で、図1の構成例の場合は4つのプログラムカウンタに対応した4つのハードウェア・スレップがあることとなる。ハードウェア・スレップは、存在する複数のスレップの中で、現在エグゼキューションされているスレップである。本プログラムで用意するハードウェア・スレップ

の数nは、任意に設定(設計)可能である。ハードウェア・スレップは、物理スレップ番号(P TID (Physical ThreadID))で管理される。

【0025】本実施形態では、詳しくは後述するように、本プロセッサにおいて各スレップに固有の仮想スレップ番号(VTID (Virtual ThreadID))を割り当てて管理し、必要に応じてスレップをハードウェア・スレップからメモリへ待避させるあるいはメモリからハードウェア・スレップへ復帰させることによって、見かけ上、ハードウェア・スレップの数を上回る数のスレップを扱うことができるようになっている。この意味におけるスレップを、ハードウェア・スレップに対して、仮想スレップと呼ぶこととする。同時に扱うことのできる仮想スレップの個数すなわち仮想スレップに割り当てることができる仮想スレップ番号(VTID)の個数mも、基本的には、任意に設定(設計)可能である。例えば、仮想スレップ番号を8ビットで表現することとした場合には、最大256個の仮想スレップ番号が使用可能となる。

【0026】さて、図1に示されるように、本プロセッサは、インストラクション・キャッシュ(Icache)1、フェッチ部(Fetch)2、n組(本例では、n=4)のインストラクション・キュー(1Q)3、n組のプログラムカウンタ(PC)13、スレップの開始アドレスを保持するn組のSPCレジスタ19、仮想スレップ番号(VTID)を保持するn組のSVTレジスタ21、イシュエ部(Issue)4、レジスタ・ファイル・キャッシュ(Register File Cache)5、コンテキスト・セーブ用メモリ11、複数の演算ユニット(本例では、2つの演算器71、72とロード・ストア・ユニット73)、データ・キャッシュ(DCache)9、メモリ・インタフェース(Memory I/F)15、1/0インタフェース(I/O Interface)17を持つ。

【0027】コンテキスト・セーブ用メモリ11は、仮想スレップに対応するデータ(例えば、対応するプログラムカウンタの内容、対応するスタックレジスタの内容等、対応するレジスタファイルの内容など;以下、コンテキストと呼ぶ)を格納する。

【0028】レジスタ・ファイル・キャッシュ5は、コンテキスト・セーブ用メモリ11に格納されている仮想スレップのコンテキストをキャッシュする。

【0029】なお、レジスタ・ファイル・キャッシュする技術としては、例えば、文献1(Peter R. Math, William J. Daily, "The Named-State Register File Implementation and Performance", First Ann. International Symposium High-Performance Computer Architecture, pp.4-13, Jan.1995)に開示されている技術を用いることができる。

【0030】また、メモリ・インタフェース15に、メモリ(Memory)23が接続される。

【0031】また、1/0インタフェース17に、例えばコプロセッサ(Coprocessor)27などが接続される。

【0032】図1に示されるように、本仮想マルチスレッププロセッサでは、個々のハードウェア・スレップ専用、プログラムカウンタ(PC)およびインストラクション・キュー(1Q)などの資源が設けられる。4つのハードウェア・スレップをサポートする場合、4組のプログラムカウンタ(PC)およびインストラクション・キュー(1Q)が必要になる。そして、ハードウェア・スレップへの仮想スレップの対応をスレップの実行に応じて適宜切り替えるようにしている。なお、本プロセッサでは、複数のスレップで演算ユニットやキャッシュなどを共有する構成にしている。

【0033】なお、図1においては接続線の一部を省略してある。

【0034】ここで、本プロセッサの一般的な動作について説明する。

【0035】インストラクション・キャッシュ1は、メモリ23から命令をキャッシュする。インストラクション・キャッシュ1がミスしたときは、メモリ23からインストラクション・キャッシュ1へ命令がリテイル(Retire)される。

【0036】データ・キャッシュ9は、メモリ23からデータをキャッシュする。データ・キャッシュ9がミスしたときは、メモリ23からデータ・キャッシュ9へデータがリテイルされる。

【0037】フェッチ部2は、命令のフェッチ(インストラクションキャッシュ1へのアタセス)を行う。

【0038】インストラクション・キュー3は、各スレップに対応する命令を保持する。

【0039】イシュエ部4は、命令の発行制御と、命令のデコードやレジスタファイルへのアタセスを行う。

【0040】演算器71、72やロード・ストア・ユニット73は、発行された命令を実行する。その際、必要に応じて所定のタイミングで、データ・キャッシュ9やレジスタファイル5に格納するアタセス(リテイル)はリテイルを行なう。

【0041】なお、このプロセッサは、パイプライン・プロセッサであってもよい。例えば、5段のパイプラインの場合、次のようなステージ構成になる。

フェッチ・ステージ…命令のフェッチ

デコード・ステージ…命令のデコード、レジスタファイルへのアタセス

実行ステージ…命令の実行

メモリ・ステージ…メモリへのアタセス

1 ライトバック(Writback)ステージ…レジスタファイルへの書き込み

次に、仮想スレップの起動に関して説明する。

【0042】まず、本実施形態では、スレップの起動には、次の2つがある。一つは、スレップが新規に起動される場合で、これを新規起動と呼ぶものとする。もう一つは、新規起動された後に一旦メモリに待避されたスレップ(後述するSleep状態からReady状態になったスレップ)が復帰されて起動する場合で、これを復帰起動と呼ぶものとする。

【0043】スレップの新規起動は、ハードウェア上で各スレップが専有する必要がある資源(プログラムカウンタおよびインストラクション・キュー等)のうち空いているものがなくとも1スレップ分存在する場合に、行うことができる(ここでは、この場合に対象スレップがあればそれを優先的に新規起動させるものとする)。例えば、図1において、ハードウェア上で3つのスレップが実行中の場合、4組のプログラムカウンタおよびインストラクション・キューなどのうち3組が使用されており、1スレップ分は使用されていないので、1つのスレップを新規起動することができ。

【0044】スレップの新規起動は、使用するハードウェア・スレップ(例えば、ハードウェア・スレップ番号PTID=0、1、2または3)に専用に設けられるSPCレジスタ19(例えば、SPC0、SPC1、SPC2またはSPC3)に、当該スレップの開始アドレスを設定しておく(なお、該開始アドレスはユーザが記述した命令の実行によって設定される)、起動命令により、該SPCレジスタ19から対応するプログラムカウンタ13(例えば、PC0、PC1、PC2またはPC3)に値を転送することによって、開始される。このとき、該ハードウェア・スレップに専用に設けられるSVTレジスタ21(例えば、SVT0、SVT1、SVT2またはSVT3)には、仮想スレップ番号を設定しておき、新規起動されたスレップは、それが起動されたハードウェア・スレップに対応するSVTレジスタ21に設定された仮想スレップ番号を持つようになる。

【0045】なお、ここでは、上記の仮想スレップ番号の割り当ておよび設定は、ソフトウェアによって(ユーザが記述した命令の実行によって)行うことにしているが、ハードウェアによって実現することも可能である。ハードウェアによって仮想スレップ番号の割り当てを自動的に行う場合には、例えば、各仮想スレップ番号について使用中か未使用であるかを知るための情報(例えば、仮想スレップ番号と使用/未使用フラグとのペアを、使用していない仮想スレップ番号のグループを管理し、スレップが新規起動する際には未使用の仮想スレップ番号から所定の基準(例えば、ランダム、番号の小さい順等)で選択したもの)を割り当ててその仮想スレップ番号を使用中として管理し、スレップが終了する際にはそのスレップが使用していた仮想スレップ番号を未使用として管理するようにすればよい。

【0046】次に、イシュエ部4の処理について説明する。

【0047】図1において、各スレップの各命令は、メモリ23から、メモリ・インタフェース15/インストラクション・キャッシュ1/フェッチ部2を経由して、スレップ対応のインストラクション・キュー3に入る。図1の例では、インストラクション・キュー3は、ハードウェア上に存在する4つのスレップに対応して4つ(1Q0、1Q1、1Q2、1Q3)ある。

【0048】イシュエ部4は、所定のイシュエ・ロジックに従って、各スレップに対応する各インストラクション・キュー3の先頭の命令の各々について、それが実行可能か否かを判断し、実行可能な命令があれば、該命令(図1の例では、最大3つ)を、該当する実行ユニット(図1の例では、71、72、73のいずれか)に送る。例えば、インストラクション・キュー1Q0の命令を、ロード・ストア・ユニット73に送り、インストラクション・キュー1Q1の命令を、演算器72に送る。

【0049】所定のイシュエ・ロジックすなわち実行可能な命令を決定するロジックとしては、例えば、その時点で、当該命令に対応する実行ユニットが空いているか、レジスタ・ファイル・キャッシュ5に対応するレジスタが存在しているか、かつ、対応するレジスタが先行する命令で書き込みが指定されている場合は、その書き込みが終了している場合に、実行可能と判断するものである。また、その際に、1以上の実行ユニットが空いており、空いている実行ユニットを使うことができる場合は、実行可能となる命令が複数ある場合に、それら命令のすべてを実行するには、空いている実行ユニットの数が不足するときは、それら命令を先頭に持つ複数のインストラクション・キュー3のうちから所定の選択基準に従って一部のものを選択され、選択されたインストラクション・キュー3の先頭の命令が、空いている実行ユニットに割り当てられる。

【0050】所定の選択基準としては、ランダムに選択する方法の他に、インストラクション・キューに着目する方法として、その時点でのキュー長が短いインストラクション・キュー3を優先的に選択する方法、予め各インストラクション・キュー3に付与された固定の優先順位に基づいて選択する方法など、様々な方法が考えられる。あるいは、スレップに割り当てる方法として、仮想スレップ番号が小さいスレップ(に対応するインストラクション・キュー)から選択する方法、各スレップについて(例えばユーザが命令を記述することにより)優先度

を設定し、より優先度の高いもの（に対応するインストラクション・キュー）から選択する方法など、種々の方法がある。また、それらを適宜組み合わせる方法も可能である。

【0051】例えば、演算器71とローフ・ストフ・ユニット73が空いており、インストラクション・キュー100の命令は、ローフ・ストフ・ユニット73で実行可能であり且つレジスタ・フイル・キャッシュ5に対応するレジスタが存在しており、インストラクション・キュー101～104の命令は、演算器71で実行可能であり且つレジスタ・フイル・キャッシュ5に対応するレジスタが存在しているものとする。ローフ・ストフ・ユニット73で実行可能な命令はインストラクション・キュー100の命令だけであるので、インストラクション・キュー101～104の命令は、演算器71に送ればよい。一方、演算器71で実行可能な命令はインストラクション・キュー101～104の3つの命令であるので、発行すべき命令（のインストラクション・キュー）の選択が必要となり、例えばインストラクション・キュー101が選択されたとすると、その命令を、演算器72に送ることになる。

【0052】次に、レジスタ・フイル・キャッシュ5の働きについて説明する。

【0053】レジスタ・フイル・キャッシュ5は、スレップに付けられた仮想スレップ番号（VTID）とレジスタ番号（RID）とをキーとして引かれるキャッシュになっている。レジスタ・フイル・キャッシュ5について、キャッシュにミシした場合は、コンテキスト・セーブ用メモリ11から必要なデータがフェッチされる。また、キャッシュの容量には、ハードウェア上の制約があるので、LRUなどのアルゴリズムなどによりフェロケートされるエンリが決定され、コンテキスト・セーブ用メモリ11に書き込まれる。

【0054】なお、レジスタ・フイル・キャッシュ5について、キャッシュ・ミシは、インシュ部4が、各スレップに対応する各インストラクション・キュー3の先頭の命令の各々について、それが実行可能か否か判断する際に、レジスタ・フイル・キャッシュ5に対応するレジスタが存在しているか否かを調べた際に発生する。この場合は、コンテキスト・セーブ用メモリ11から必要なデータがフェッチされるので、インシュ部4が当該命令について次に判断を行う際には、レジスタ・フイル・キャッシュ5には対応するレジスタが存在している状態になっている。

【0055】一方、実行ユニット（例えば71等）の演算結果やメモリ3からデータ・キャッシュ9経由で読み出されたデータは、レジスタ・フイル・キャッシュ5に書き戻される。

【0056】なお、プロセッサ時にコンテキスト・セーブ用メモリ11に書き込まなくてもよい場合を抜出す

るため、ソフトウェアからdeadなレジスタを示す手法を用いてもよい。この手法によれば、deadなレジスタは、コンテキスト・セーブ用メモリ11に書き込む必要がなく、新たなレジスタが必要になった場合のプロセッサの速度を早くすることができる。この手法としては、例えば、文献2[ack Llo et al, "Software-Threaded Register Deallocation for Simultaneous Multithreaded Processors", IEEE Trans. on Parallel and Distributed Systems, Vol.10, No.9, Sep, 1999]に開示された技術を用いることができる。次に、仮想スレップの状態およびその遷移について説明する。

【0057】図2に、仮想スレップの4つの状態と、状態間の遷移関係を示す。

【0058】仮想スレップの状態には、バイフアインション・キュー3に於ける実行できない状態（Wait状態）、同期など応答を待っている状態（Sleep状態）、Sleep状態から応答が返って実行可能な状態（Ready状態）の4つがある。

【0059】状態間の遷移としては、Active状態とWait状態との間の遷移関係と、Active状態からSleep状態へ／Sleep状態からReady状態へ／Ready状態からActive状態への遷移関係との2系統がある。

【0060】Active状態からWait状態への遷移は、キャッシュ・ミシやレナシなどによって起こり、それが解消することによって、Active状態からWait状態へ戻る。

【0061】Active状態からSleep状態への遷移は、例えば命令のレナシが実行される場合、例えばコンテキストの場合（あるいは他の理由でスリープすべき場合）などにより、明示的に行われる。

【0062】Sleep状態からReady状態への遷移は、Sleep状態に遷移するときに出した命令あるいは要求（例えばあるコンテキストにある処理の実行を要求する命令）に対応する応答が、（例えば要求を受けたコンテキストから）返されたことを契機として行われる。なお、この他に、あるスレップがSleep状態の他のスレップをReady状態にすることを指示する命令を用意することもある。

【0063】Active状態のスレップがSleep状態に遷移すると、いままでインストラクション・キュー3に命令を結んでいた当該スレップに対応するプログラムカウンタ13は無効化される（インストラクション・キュー3内の命令も無効化される）。なお、この時点で、レジスタ・フイル・キャッシュ5の該当するデータを維持しておく方法と、コンテキスト・セーブ用メモリ11に追いつき方法とがある。

【0064】次に、Ready状態からActive状態への遷移について説明する。

【0065】上記のように、Active状態のスレップがSleep状態に遷移して、対応するプログラムカウンタ13は無効化されると、新規起動または復帰起動が可能になる。ここでは、新規起動できるスレップがなければ新規起動を行い、新規起動できるスレップがなく且つ復帰起動できるスレップがあれば復帰起動するものとする。なお、新規起動できるスレップも復帰起動できるスレップもなければ、新規起動できるスレップまたは復帰起動できるスレップが出現するまで、当該ハードウェア上リソースは未使用（空き状態）になる。

【0066】さて、Ready状態のスレップを復帰起動できることとなった場合には、本実施形態では、システムバスと呼ばれる特殊なスレップによって、Ready状態のスレップが複数ある場合におけるスレップの選択と、スレップの復帰起動のための処理を行うものとする。このメカニクスにより、実行中のスレップの邪魔をしないことを可能にしている。

【0067】この場合、前述のようにして無効化されたプログラムカウンタ13には、新たにシステムバスの番地がセットされる。

【0068】起動されたシステムバスは、Ready状態になっているスレップのうちから1つを選択し、そのスレップのプログラムカウンタの値を、対応するプログラムカウンタ13にセットするなどの処理を行う。すると、該プログラムカウンタの値に基づいて、命令が新たにフェッチされ始め、対応するインストラクション・キュー3に、命令が入れられる。

【0069】なお、Ready状態になっているスレップが複数ある場合には、システムバスが、それから1つを選択するアルゴリズムには、様々なものが考えられる。例えば、ランダムに選択する方法、FIFOでもっとも古くReady状態になったものを選択する方法、仮想スレップ番号が小さいものから選択する方法、各スレップについて（例えばユーザが命令を記述することにより）優先度を設定し、より優先度の高いものを選択する方法など、種々の方法がある。

【0070】なお、あるスレップがReady状態の他のスレップをActive状態にすることを指示する命令を用意することも可能である。

【0071】ここで、Ready状態のスレップ番号に対応して、Ready状態の場合にReadyピットに1が立つ（それ以外の場合にはReadyピットは0とする）ようにしたReadyレジスタを設ける方法がある（なお、Sleep状態（例えば、01）か、Ready状態（例えば、10）か、またはそれ以外の状態（例えば、00）かを示すようにする方法もある）。システムバスでは、Readyレジスタを参照し、Readyピットに1（あるいは、10）が立っているスレップを探索し、それが1つあればそのスレップを、複数あれば

はそれから選択した1つのスレップを、ハードウェア・スレップにする。

【0072】なお、本実施形態では、要求（例えばコンロセッサ命令）およびその応答には、仮想スレップ番号を付加するものとする。すなわち、要求を出す際には、その要求元のスレップの仮想スレップ番号を要求に付加するものとする。応答については、例えば、要求を受けた資源が応答を出す際に、該要求に付加されていた仮想スレップ番号を該応答に付加する。また、例えば、要求／応答と仮想スレップ番号との対応を管理する管理部を設け、要求を出す際には、その要求元のスレップの仮想スレップ番号を該要求に付加して一旦管理部に渡し、管理部が仮想スレップ番号の付加されていない要求を資源に渡し、該資源は仮想スレップ番号の付加されていない応答を返し、管理部は、この応答を受け、対応する仮想スレップ番号を特定する、という構成をとることも可能である。このようにすれば、応答が返ってきた場合（例えば、コンロセッサ命令の実行などでSleep状態になっているスレップに、応答が仮想スレップ番号（VTID）とともに返ってくるので、この仮想スレップ番号によってReadyレジスタの対応するピットを立てるだけである。

【0073】なお、上記では、仮想スレップ番号に対応してReadyピットを持つReadyレジスタを設ける方法を示したが、Sleep状態またはReady状態のスレップの仮想スレップ番号とその状態を示す情報（例えば、Sleep状態のとき0、Ready状態のとき1）とを1つのエンリとして含むReadyテーブルを設けることも可能である。

【0074】ここで、ReadyレジスタのReadyピットが、Sleep状態か、Ready状態か、またはそれ以外の状態（null状態と呼ぶ）（例えば、起動されたが特連中ではない状態、あるいは、まだ新規に起動されていない状態（スレップが終了して該仮想スレップ番号が解放された状態を含む））をとる場合（あるいはReadyレジスタ（あるいは、Readyテーブル）のReadyピットが、Sleep状態か、またはReady状態をとる場合）の処理手順について説明する。

【0075】図3に、Sleep状態に遷移する場合として、コンロセッサ命令などのレナシの長い命令が実行される際の処理手順の一例を示す。

【0076】ある仮想スレップ番号のスレップについてコンロセッサ命令などのレナシの長い命令が実行されたときは、Readyレジスタ（あるいは、Readyテーブル）の当該Sleep状態を示す値に設定した（スレップS11）、該当するハードウェア・スレップを無効化する（スレップS12）。なお、スレップS11とスレップS12は、上記とは逆の順番で行ってもよい。

並列的に行ってもよい。

【0077】図4に、応答を受けた場合の処理手順の一例を示す。

【0078】応答があった場合、応答に伴って送られる返信スレド番号から、Readyレジスタ（あるいは、Readyチーザ）を検索する（ステップS21）。Readyレジスタ（あるいは、Readyチーザ）において、応答に伴って送られた返信スレド番号と同じ返信スレド番号を持つスレド番号であってそのReadyビットがSleep状態を示すものがあれば（ステップS22）、該ReadyビットをReady状態を示す値に変更する（ステップS23）。

【0079】なお、応答に伴って送られた返信スレド番号がいずれのチーザにも登録されていない場合には、何もしないこととするか、または所定のエラー処理をすることとする（ステップS24）。

【0080】なお、応答があつてから、対応するスレドが復帰するまでの間、応答（に関するチーザ；例えば、戻り値などを含む）は、返信スレド番号に対応付けて保持される。

【0081】図5に、システムタスクに関する処理手順の一例を示す。

【0082】Readyレジスタ（あるいは、Readyチーザ）にReadyビットがReady状態を示すスレド番号がある場合に、ハードウェア・スレドに空きがあり、かつ、新規起動すべきスレドがなければ、当該空きハードウェア・スレドでシステムタスクが起動される（ステップS31）。起動されたシステムタスクは、Ready状態のスレドが1つならばそのスレドを復帰させるものとし、Ready状態のスレドが複数あるならば復帰させるものとして1つのスレドを選択し、その復帰させるスレドの返信スレド番号（VTID）に基づいて、プログラマカウンタの値の復帰などの処理を行い、復帰起動させる（ステップS32）。なお、Readyレジスタでは、復帰されたスレドに対応するReadyビットがnull状態にされ、Readyチーザでは復帰されたスレドに対応するエントリが無効にされる。

【0083】これら返信スレドに関する処理は、イシエ一部4が行うようにしてもよい、イシエ一部4以外の制御部を設けてもよい。

【0084】以上説明したように、本発明によれば、スレドの待機・復帰を可能とすることによって、プログラマにハードウェアで用意されたより多くのスレドを見ることが可能になり、プログラミングの自由度が増す。また、その際のスレドの切り替えも、効率的かつ高速に行うことができる。

【0085】また、上記では、システムタスクをスレドで実現したが、システムタスク専用のハードウェアで構成することも可能である。

【0086】また、これまでの説明では、Ready状態のスレドは、ハードウェア・スレドに空きが存在し、かつ、新規起動すべきスレドがない場合に、復帰起動することもできるものとしたが、新規起動すべきスレドがあつても、Ready状態のスレドを復帰起動することを可能としてもよい。例えば、ハードウェア・スレドに空きが存在するときに、1または複数の新規起動すべきスレドと、1または複数のReady状態のスレドがあつた場合に、予め定められた選択基準に基づいて選択した1つの新規起動すべきスレドについてはReadyビットを新規起動または復帰起動させるようにしてもよい。例えば、各スレドについて（例えばユーザが命令を記述することにより）優先度を設定し、より優先度の高いものを選択する方法、新規起動すべきスレドの数とReady状態のスレドの数とに基づいて選択する方法、Ready状態のスレドの数が所定数を超えた場合にのみReady状態のスレドを優先して選択する方法など、種々の方法がある。

【0087】また、これまでの説明は、SMT（Simultaneous MultiThread）プロセッサの例として、In-Order実行ベースのプロセッサを中心として行ったが、本発明はOut-Of-Orderプロセッサにも適用可能である。

【0088】（第2の実施形態）次に、本発明の第2の実施形態に係るスレド間の同期方法について説明する。

【0089】以下で説明するスレド間の同期方法は、実行時に資源の割り当てをダイナミックに行う第1の実施形態の返信スレドチーザ（または、第1の実施形態の返信スレドチーザの構成のうち本実施形態によって必要な部分のみ備える返信スレドチーザ）に適用した場合を例にとつて説明するが、本発明は、その他、第1の実施形態のOut-Of-Orderにした場合の返信スレドチーザ、コンパイル時にスタックに資源を割り当て使用するVLIW（Very Long Instruction Word）の返信スレドチーザ、それらにおいてレジスタ・フレイムをハードウェア・スレド毎に持つようにしたものなど、種々の返信スレドチーザに対しても適用可能である。

【0090】さて、マルチスレドのプログラマにおいては、複数のプログラマが終了したことを互いに知ることが必要となる。例えば、並列に実行しているスレドAとスレドBとスレドCの結果をスレドDが使う場合に、スレドAとスレドBとスレドCは同期を取らなければならない。本実施形態では、このような場合に、SYNC命令（同期命令）を使うものとする。SYNC命令には、カウンタを操作する機能と、同期をとるための機能とが含まれる。

【0091】本実施形態では、返信スレド番号（VTID）ごとに同期機構を実現するためのカウンタを設けるものとする。

【0092】今、スレドFAの返信スレド番号（VTID）を0、スレドFBの返信スレド番号（VTID）を1、スレドFCの返信スレド番号（VTID）を2とし、同期のためのカウンタとして、スレドFAのVTID=0に対応するカウンタを使用するものとする（いずれのスレドに対応するカウンタを使用するかは、ユーザが任意に決めて構わない）。このカウンタをSYNCカウンタ（同期カウンタ）と呼ぶ。SYNCカウンタの初期値としては、スレドFAが、SYNCを行うスレドの値3に対応して、3を設定しておく（例えば、ユーザが、同期をとるスレドの数を設定する命令を記述する）。

【0093】まず、最初の例では、スレドFAが最後にSYNC命令に到達した場合を示す。

【0094】図6で示すように、最初にSYNC命令に到達したスレドFCは、SYNCカウンタをデクリメントし、その値を2とする。SYNCカウンタの値が0でないので、スレドFCはSleep状態に遷移する。

【0095】次にSYNC命令に到達したスレドFBは、SYNCカウンタをデクリメントして、その値を1とする。また、SYNCカウンタは0でないので、スレドFBはSleep状態に遷移する。

【0096】最後にSYNC命令に到達したスレドFAでは、SYNCカウンタをデクリメントし、その結果が0であるので、他のスレドを起すためのルーチンを実行する。このルーチンでは、スレドBおよびスレドCのReadyレジスタ（あるいは、Readyチーザ）のReadyビットを1に立てること、Sleep状態からReady状態へ遷移させる。

【0097】Ready状態に遷移した後は、例えば、（条件が整えば）システムスレドによってハードウェア・スレドに復帰することができ。

【0098】SYNC命令は、次の2つの情報のフレイムを持つものとする。

【0099】一つの情報は、SYNCカウンタを識別する識別子で、これには返信スレド番号（VTID）を付加した、なお、複数のスレドが実行し、複数の同期ポイントと同時に設定する可能性があることを想定する場合に、SYNCカウンタを複数設ける。

【0100】もう一つの情報は、SYNC命令を実行した後、SYNCカウンタが0になった場合に実行するルーチンの先頭番地である。上記の例では、例えば、スレドFAとスレドBとスレドCのうちのSleep状態のものをReady状態にするルーチンの先頭番地である。

【0101】SYNCカウンタは、必ずしもハードウェア

で常に存在する必要はなく、先に説明したレジスタ・フレイム・キータン5に入れられてもよい。すなわち、SYNCカウンタは、返信スレド番号ごとに存在しているため、返信スレド番号とSYNCカウンタを示すレジスタ番号とをキーとすれば、通常のレジスタと全く同じには読み出すことが可能となり、コンテキスト・セーブ用メモリ11に必要に応じてセーブもしくはリストアされる。

【0102】図7に、スレドFAがSYNC命令に到達した場合の処理手順の一例を示す。

【0103】SYNC命令に到達したならば、まず、SYNCカウンタをデクリメントする（ステップS41）。

【0104】ここで、デクリメントした後のSYNCカウンタの値が0でなければ（ステップS42）、当該スレドを、Sleep状態に遷移させる（ステップS44）。

【0105】一方、デクリメントした後のSYNCカウンタの値が0であれば（ステップS42）、同期をとる関係にある全スレドのうちSleep状態にあるものをReady状態にするためのルーチンを実行する（ステップS43）。Ready状態に遷移した後は、例えばシステムスレドによって復帰される。

【0106】なお、これらのスレド間の同期に関する処理は、例えば図1の構成の場合には、イシエ一部4が行うようにしてもよい、イシエ一部4以外の制御部を設けてもよい。

【0107】本発明を用いないデジタールウェアでは、CPUが無駄に消費されるのみならず、デジタールウェアの箇所がハードウェアスレドの数に等しいか多ければデジタールウェアが起る。また、割り込みを用いる方法では、オーバヘッドが大きいの。これに對して、本発明のマルチスレド間の同期方法を用いれば、それら問題を回避して、効果的な同期を行うことができる。

【0108】なお、本実施形態で例示した構成は一例であつて、それ以外の構成を排除する趣旨のものではなく、例示した構成の一部を他のもので置き換へたり、例示した構成の一部を省いたり、例示した構成に別の機能を付加したり、それらを組み合わせたことなどによって得られた別の構成も可能である。また、例示した構成と論理的に等価な別の構成、例示した構成と論理的に等価な部分を含む別の構成、例示した構成の要部と論理的に等価な部分を含む別の構成、例示した構成と同一もしくは類似の目的を達成する別の構成、例示した構成と同一もしくは類似の効果を奏する別の構成なども可能である。また、各種構成部分についての各種バリエーションは、適宜組み合わせて実施することが可能である。また、本実施形態は、装置としての発明、装置内部の構成部分についての発明、またはそれらに對する発明の方法の発明等、種々の観点、態様、またはカ

メモリに係る発明を包含・内在するものである。従って、この発明の実施形態に開示した内容からは、例示した構成に限定されることなく発明を抽出することができるものである。

【0109】本発明は、上述した実施形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【0110】

【発明の効果】本発明によれば、レジスタ・フイルを、仮想スレップ識別番号およびレジスタ識別番号をキ一とするキヤンシユとして構成することで、プロセッサ上の資源に割り当てた仮想スレップ識別番号を適宜切り替えることによって、実際にハードウェア上に存在するスレップより、仮想的に多くのスレップが実行されているようにみせることができる。これによって、プログラマにハードウェアで用意されたより多くのスレップを見せることが可能になり、プログラミングの自由度が増す。

【0111】また、本発明によれば、同期機構を実現するためのカウンタを設け、同期にかかわる複数のスレップが同期命令によってこのカウンタをデクリメントすることによって、効率的にスレップ間の同期をとることができる。

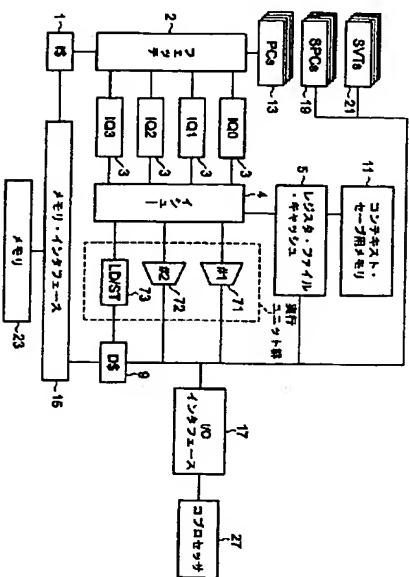
【図面の簡単な説明】

【図1】本発明の一実施形態に係る仮想マルチスレッププロセッサの構成例を示す図

【図2】同実施形態における仮想スレップの状態およびその状態間の遷移関係について示す図

【図3】同実施形態におけるレーテンジの長い命令が実行される際の処理手順の一例を示すフローチャート

【図1】



*行される際の処理手順の一例を示すフローチャート

【図4】同実施形態におけるレーテンジの長いハードウェア資源への要求に対する応答を受けた場合の処理手順の一例を示すフローチャート

【図5】同実施形態におけるシステムスタックに関する処理手順の一例を示すフローチャート

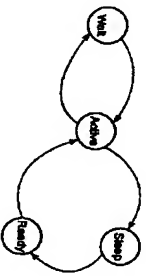
【図6】同実施形態におけるスレップ間の同期方法について説明するための図

【図7】同実施形態におけるスレップがSYNC命令に到達した場合の処理手順の一例を示すフローチャート

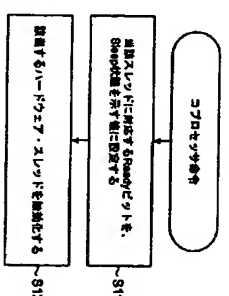
【符号の説明】

- 1...レジスタ・フイル・キヤンシユ
- 2...レジスタ・フイル・キヤンシユ
- 3...レジスタ・フイル・キヤンシユ
- 4...レジスタ・フイル・キヤンシユ
- 5...レジスタ・フイル・キヤンシユ
- 6...レジスタ・フイル・キヤンシユ
- 7...レジスタ・フイル・キヤンシユ
- 8...レジスタ・フイル・キヤンシユ
- 9...レジスタ・フイル・キヤンシユ
- 10...カウンタ
- 11...同期機構
- 12...レジスタ・フイル・キヤンシユ
- 13...レジスタ・フイル・キヤンシユ
- 14...レジスタ・フイル・キヤンシユ
- 15...レジスタ・フイル・キヤンシユ
- 16...メモリ・インタフェース
- 17...メモリ
- 18...メモリ
- 19...メモリ
- 20...メモリ
- 21...メモリ
- 22...メモリ
- 23...メモリ
- 24...メモリ
- 25...メモリ
- 26...メモリ
- 27...メモリ
- 28...メモリ
- 29...メモリ
- 30...メモリ
- 31...メモリ
- 32...メモリ
- 33...メモリ
- 34...メモリ
- 35...メモリ
- 36...メモリ
- 37...メモリ
- 38...メモリ
- 39...メモリ
- 40...メモリ
- 41...メモリ
- 42...メモリ
- 43...メモリ
- 44...メモリ
- 45...メモリ
- 46...メモリ
- 47...メモリ
- 48...メモリ
- 49...メモリ
- 50...メモリ
- 51...メモリ
- 52...メモリ
- 53...メモリ
- 54...メモリ
- 55...メモリ
- 56...メモリ
- 57...メモリ
- 58...メモリ
- 59...メモリ
- 60...メモリ
- 61...メモリ
- 62...メモリ
- 63...メモリ
- 64...メモリ
- 65...メモリ
- 66...メモリ
- 67...メモリ
- 68...メモリ
- 69...メモリ
- 70...メモリ
- 71...メモリ
- 72...メモリ
- 73...メモリ
- 74...メモリ
- 75...メモリ
- 76...メモリ
- 77...メモリ
- 78...メモリ
- 79...メモリ
- 80...メモリ
- 81...メモリ
- 82...メモリ
- 83...メモリ
- 84...メモリ
- 85...メモリ
- 86...メモリ
- 87...メモリ
- 88...メモリ
- 89...メモリ
- 90...メモリ
- 91...メモリ
- 92...メモリ
- 93...メモリ
- 94...メモリ
- 95...メモリ
- 96...メモリ
- 97...メモリ
- 98...メモリ
- 99...メモリ
- 100...メモリ

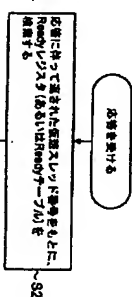
【図2】



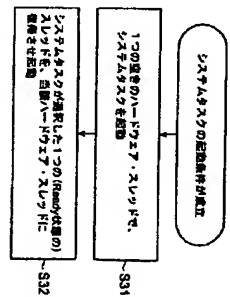
【図3】



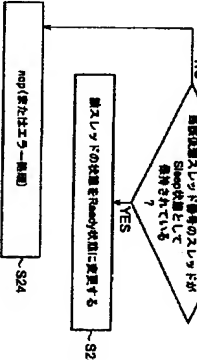
【図4】



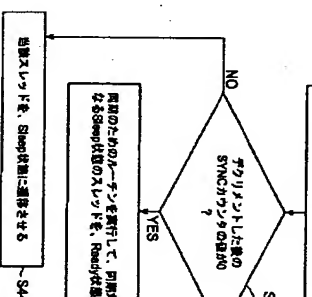
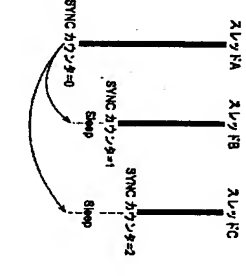
【図5】



【図7】



【図6】



フロントページの抜き

フォーマット(参考) SB005 LL11 MM01

SB098 F01 GA05 GC01 GN02 GD05

GD14

THIS PAGE BLANK (USPTO)